AQA Computer Science GCSE
3.4 Computer systems
Advanced Notes

This work by PMT Education is licensed under CC BY-NC-ND 4.0

3.4.1 Hardware and software

What is hardware?

Hardware refers to the physical components of a computer system - the parts you can see and touch.

Examples:

- CPU (Central Processing Unit)
- RAM (Random Access Memory)
- Hard drive or SSD
- Keyboard, mouse, monitor
- Motherboard, graphics card

What is software?

Software is the program code that is executed by the hardware. It is a set of instructions that controls the operations of the hardware.

The relationship between software and hardware

Software needs hardware to execute and function on. Hardware can't function without software to control it and tell it what to do.

3.4.2 Boolean logic

What is Boolean logic?

Boolean logic is a form of algebra where all values are either TRUE (1) or FALSE (0). It is used in computers to control decision-making and logical operations.

Computers use Boolean logic in:

- Programming conditions
- Logic gates and circuits
- Searching and filtering

Truth tables

A truth table is used to show the output of Boolean expressions for all possible input combinations. This means that they are useful for debugging and understanding logic conditions.

Logic gates

A computer's processor is made up of billions of logic gates, devices which apply logical operations to one or more Boolean inputs in order to produce a single output.

Within a processor, logic gates are combined to form logic circuits. These can perform more complex operations like binary addition.

AND

- Returns **TRUE** only if **both inputs** are TRUE
- Symbol: `A AND B`, `A . B`

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

OR

- Returns **TRUE** if **either** input is TRUE
- Symbol: `A OR B, A + B`

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

XOR

- Returns **TRUE** if **only one** input is TRUE
- Symbol: `A XOR B,` $A \oplus B$

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

NOT

- Reverses the input
- Symbol: `NOT A,` $\overline{A}$

A	NOT A
0	1
1	0

www.pmt.education PMTEducation

Combining operators

You can combine logic gates and build expressions like:

`A AND (B OR NOT C)`, $A \cdot (B + \overline{C})$

Order of precedence

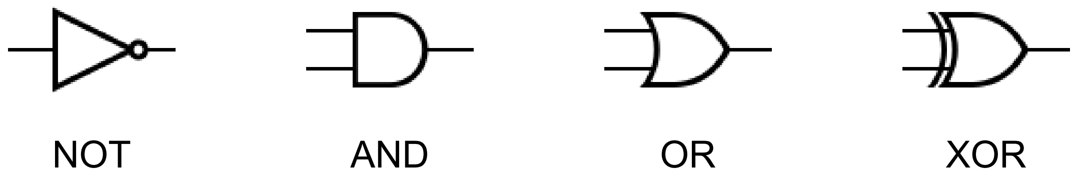
Algebraic operations have an order of precedence, meaning that some operations must be applied before others. You may have met BODMAS in Mathematics, this is the same idea.

Operator	Precedence
Brackets	Highest
NOT	.
AND	.
OR	Lowest

For example, the expression `B OR NOT C AND A` would actually be carried out in the order `B OR ((NOT C) AND A)`.

Logic circuit symbols

Each of the four required logic gates has an internationally recognised symbol which you should learn. The symbols have inputs on the left and outputs on the right.

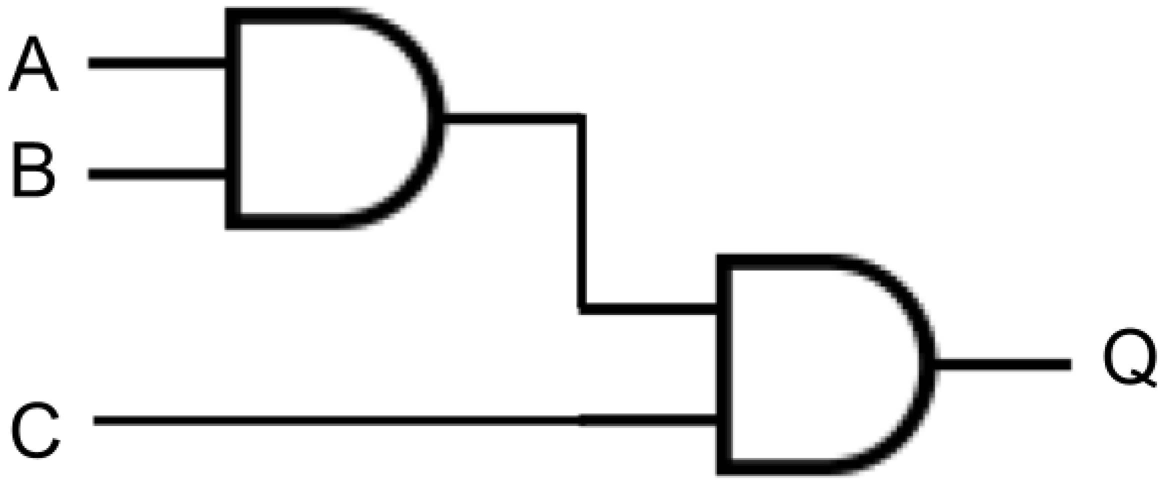


These symbols can be combined to form logic circuit diagrams.

Logic circuit diagrams

Logic gates can be combined to form more complex circuits. You may be asked to draw or interpret a logic circuit involving multiple logic gates.

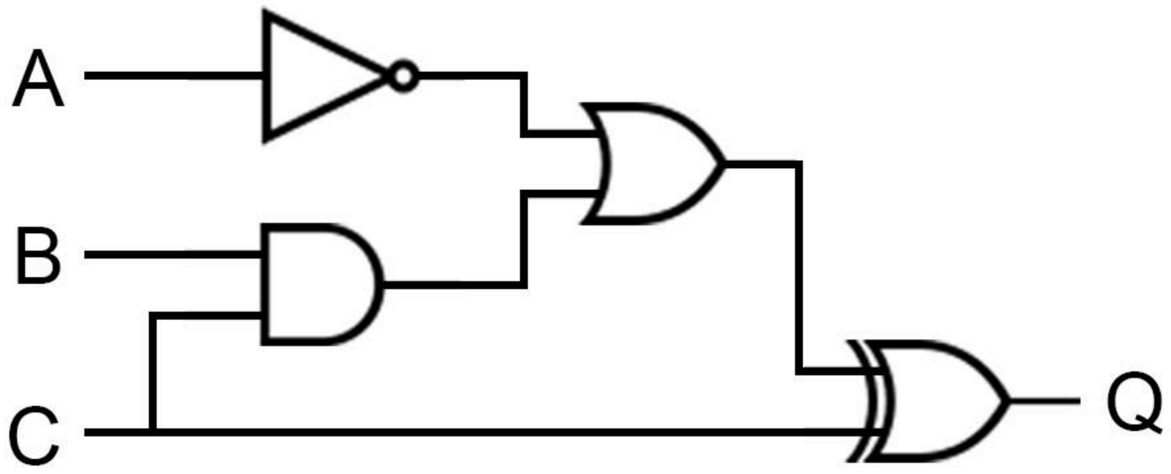
Example 1 - combining two AND gates



In this example, the output Q is TRUE only if A, B AND C are all TRUE.

The Boolean expression from this logic circuit is A AND B AND C.

Example 2 - a more complex circuit

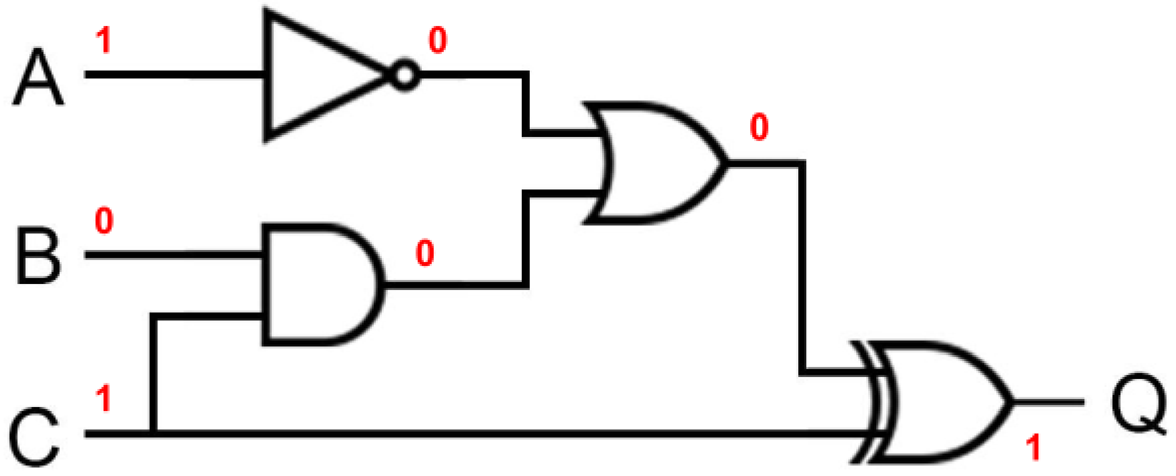


In this example, A, B and C are the inputs, which could each be 0 or 1. Q is the output - representing the result of the logic gates in their specific arrangement being applied to the inputs.

If we were given the following inputs:

A	B	C
1	0	1

We can calculate the output, Q by writing the result of each logic gate next to it:



This gives the result that Q is TRUE (1).

Boolean expression operators

In addition to logic gate symbols, the logic gates can also be represented using Boolean expression symbols:

Gate	Symbol
AND	**.** (dot)
OR	**+**
XOR	⊕
NOT	$\overline{Overbar}$

For example the expression (A AND B) OR (NOT C) would be represented as:

$$(A.B) + \overline{C}$$

3.4.3 Software classification

Software can be classified into two types:

1. System software
2. Application software

1. System software

System software manages and controls the computer hardware and acts as a platform to run application software.

Examples:

- Operating Systems (OS): e.g. Windows, macOS, Linux
  The operating system handles management of the:
  - Processor(s)
  - Memory
  - Input/output (I/O) devices
  - Applications
  - Security
- Utility programs: e.g. antivirus, backup software
  Utility software are programs designed to help maintain, enhance, and troubleshoot a computer system
- Device drivers: allow OS to communicate with hardware
- Firmware: built-in software controlling hardware (e.g. BIOS)

2. Application software

Application software that performs specific tasks for the end-user.

Examples:

- Word processors (such as Microsoft Word)
- Web browsers (such as Chrome, Firefox)
- Games, media players, email clients

3.4.4 Classification of programming languages and translators

Programming languages are used to write instructions that computers can execute. They fall into two main categories:

- High-level languages (e.g. Python, Java, C#)
- Low-level languages (e.g. Assembly language, Machine code)

High-level languages

Designed for humans to read and write, with instructions that are similar to English (such as `print` and `while`). Most computer programs are written using high-level languages. Examples: Python, C#.

Advantages	Disadvantages
Easy for humans to understand and debug as the instructions are closer to English	Slower to execute than low-level languages
Programs written are portable between different hardware, since they can be translated into machine code for each specific type of processor	Must be translated into machine code, and this translated machine code can be less efficient than if it was originally written as machine code.

Low-level languages

Closer to machine code (binary). Examples: Assembly language, Machine code

Advantages	Disadvantages
Faster and more efficient to execute	Hard to read and write
Gives more control over hardware	Not portable - specific to one type of processor

Translators

Computers only understand machine code, so all programs written in high-level or assembly languages must be translated before they can be executed. Machine code is expressed in binary and is specific to a processor or family of processors.

Types of translators

There are three types of program translator: assemblers, compilers and interpreters.

Assemblers

An assembler translates assembly language into machine code. Because each assembly language instruction has a 1:1 relationship to a machine code instruction, translation between the two languages is fairly quick and straightforward. Assembly language is often used to develop software for embedded systems and for controlling specific hardware components. Assemblers are platform specific, meaning that a different assembler must exist for each different type of processor instruction set.

Compliers

A compiler can be used to translate programs written in high-level languages like C# and Python into machine code. Compilers take a high-level program as their source code, check it for any errors and then translate the entire program at once. If the source code contains an error, it will not be translated. Because compilers produce non-portable machine code, they are said to be platform specific.

Once translated, a compiled program can be run without the requirement for any other software to be present. This is not the case with interpreters.

Interpreters

An interpreter translates high-level languages into machine code and executes it line-by-line. Interpreters do not generate machine code directly - they call appropriate machine code subroutines within their own code to carry out statements.

Rather than checking for errors before translation begins (as a compiler does), interpreters check for errors as they go. This means that a program with errors in can be partially translated by an interpreter until the error is reached.

When a program is translated by an interpreter, both the program source code and the interpreter itself must be present. This results in poor protection of the source code compared to compilers which make the original code difficult to extract.

Comparison of compilers and interpreters

Compiler	Interpreter
Checks source code for errors line-by-line before beginning translation	Translation begins immediately
Entire source code translated at once	Each line is checked for errors and then translated sequentially
No need for source code or compiler to be present when the translated code is executed	Both the source code and the interpreter must be present when the program is executed
Protects the source code from extraction	Offers little protection of source code

3.4.5 Systems architecture

Purpose of the CPU

At the heart of every computer is a Central Processing Unit (CPU) which executes instructions in order to run programs. Processors contain an arithmetic logic unit, a control unit and numerous registers.

Major CPU components

Component	Function
Arithmetic logic unit	Performs mathematical calculations and logical operations as required.
Control unit	The control unit controls the operation of the fetch-decode-execute cycle, synchronising the operation of the CPU and sending commands to other components - for instance, requesting that the arithmetic logic unit perform a calculation - via the buses.
Clock	A device that sends a regular electrical signal which changes at a regular frequency. This signal is used to synchronise the computer system's components; it influences the number of instructions carried out each second.
Registers	Fast-to-access storage locations, used to store small amounts of data needed temporarily by the CPU during processing, such as the current instruction being decoded or the result of calculations performed by the arithmetic logic unit.
Bus	A collection of wires through which data/signals are transmitted from one component to another.

CPU performance factors

Factor	Description
Clock speed	With every tick of the clock, the CPU fetches and executes one instruction. The greater the clock speed, the faster the CPU can execute instructions.
Cache size	Cache is a small, fast memory device located on the CPU that stores frequently used data and instructions. A larger cache reduces the need for the CPU to access slower main memory (RAM) as often, improving performance.
Number of cores	Cores are the individual processing units within a CPU. More cores enable the CPU to handle multiple tasks in parallel (simultaneously) provided the software being used allows it, making it faster.

The fetch-decode-execute cycle

The fetch-decode-execute cycle is a continuous cycle performed by the processor. It consists of three stages: fetch, decode and execute. This cycle repeats millions of times per second.

1. **Fetch**: The next instruction is fetched from memory to the CPU.

2. **Decode**: The control unit interprets the fetched instruction to determine what operation needs to be performed.

3. **Execute**: The instruction is carried out. Data required by the instruction may be fetched from main memory, or the instruction may store a value in main memory.

Von Neumann architecture

Most modern CPUs follow the Von Neumann model, where instructions and data are stored in the same memory**,** and one bus is used to transfer both.

Different types of memory within a computer

Memory (or primary storage) is where data and instructions are stored for quick access by the CPU during program execution.

Memory type	Description
RAM	RAM stands for Random Access Memory, and it is a form of main memory. RAM holds the data and instructions that the computer is currently working with, such as the operating system, running applications, and open documents. RAM is volatile, meaning its contents are lost when the computer loses power (e.g., when turned off).
ROM	ROM stands for Read-Only Memory, and it is a form of main memory. It is typically used to store firmware that is essential for the computer to boot up and operate. As the name suggests, it is read-only; it cannot be written to or modified during normal operation. It is also non-volatile, meaning that it retains its contents even when the power is off.
Cache	The cache is fast-access memory located directly on the CPU, and it stores frequently used data/instructions. This makes receiving data/instructions from the computer's memory more efficient.
Register	Fast-to-access storage locations, used to store small amounts of data needed by the CPU during processing, such as the current instruction being decoded or the result of calculations performed by the arithmetic logic unit.

Differences between main memory and secondary storage

Main memory encompasses all memory directly accessible by the CPU, excluding cache and registers. This includes RAM (Random Access Memory) and ROM (Read-Only Memory).

Secondary storage is considered to be any non-volatile storage mechanism not directly accessible by the CPU. Secondary storage is needed so that data/files can be stored on a long-term basis, using non-volatile storage so that they are retained when the computer is switched off.

Types of secondary storage

Three types of secondary storage are solid-state, optical and magnetic.

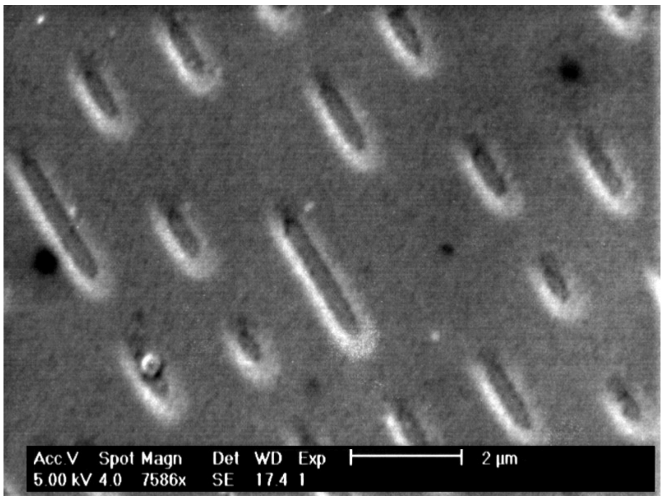
Solid-state

Solid State Drives (SSDs) use electrical circuits to persistently store data. They don't have any moving parts, so are capable of far higher read and write speeds than magnetic HDDs and are suitable for use in portable devices like phones and tablets.

Optical

Optical disks include CDs, DVDs and Blu-rays. They store information which can be read optically by a laser.

The image below shows a microscope view of the surface of a read-only optical disk. The stripes in the image are called pits, and the areas surrounding them are called lands. Pits are burnt into the disk by a high-power laser which permanently deforms the surface.

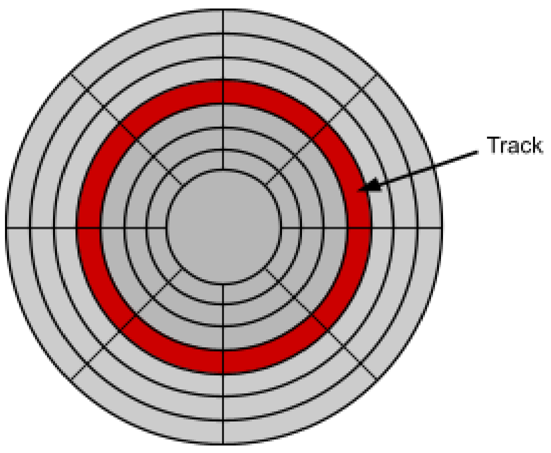


When a laser is shone at the disk, the intensity of the reflected light is measured. The continuation of a land/pit reflects light (representing a 0) whereas a transition between a land and a pit scatters light (representing a 1).

www.pmt.education PMTEducation

Magnetic

On a magnetic hard disk, binary data is represented by tiny magnetised regions, where the magnetic orientation in one direction represents 0, and the other direction represents 1. Data is written in concentric tracks, each of which is further divided into sectors.



When reading data the read/write head is moved to be over the correct track, and the disk spins round. A whole sector is read in one go by the read/write head.

Comparison of secondary storage devices

	Hard-disk drive	Solid-state drive	Optical disk
Capacity	High capacity.	Relatively low capacity.	Very low capacity.
Read / write speeds	Good speeds.	Very high speeds.	Relatively low speeds.
Portability	Bulky, heavy and easily damaged by movement.	Lightweight and rarely damaged by movement.	Very small and lightweight, can be damaged by scratches and dirt.
Durability	Contains moving parts, prone to damage	No moving parts, very durable.	Easily scratched or damaged.
Reliability	Fairly reliable but degrades over time.	Very reliable.	Less reliable - damage affects data easily.
Cost	Cheap per GB.	Expensive per GB.	Very cheap per disk, but poor cost per GB.
Suitability	Good for desktop PCs and servers.	Good for laptops, phones and tablets.	Good for sharing and distributing small volumes of data.

Cloud storage

Cloud storage allows users to store their files in a remote location, where magnetic and/or solid state storage is used to store their files on their behalf. Several companies such as Dropbox and Google Drive offer cloud storage as a service.

Advantages compared with local storage

- Enables users to access their data from more places and devices
- Parts of cloud storage can be made publicly available to others, enabling users to share their data more easily
- The cost of computing devices can be made cheaper to users as there is no need for as much built-in secondary storage

Disadvantages compared with local storage

- Cloud storage could potentially cost more in the long-term, as costs are typically ongoing
- There are potential data privacy issues, as there is an increased chance of others accessing personal data
- Relies on access to high-bandwidth network connection

Embedded systems

An embedded system is a computer system that is designed to perform specific, dedicated functions within a larger mechanical or electronic system. It is "embedded" into a device to control particular operations of that device.

How embedded systems differ from non-embedded systems

Embedded Systems:
- Designed for one specific task or set of related tasks
- Built into other devices and cannot easily be separated
- Have minimal or no user interface
- Optimised for efficiency and reliability

Non-Embedded Systems (General-purpose Computers):
- Can run many different applications and programs
- Standalone systems (like desktop computers, laptops)
- Have comprehensive user interfaces
- Software can be easily installed, removed, or updated
- Users can multitask between different applications
- Highly upgradeable and customisable
- Optimised for flexibility and performance

Examples

Embedded Systems:

- Washing machines - embedded systems can control water temperature, cycle timing and motor speed
- Microwave ovens - embedded systems can manage cooking time, power levels and safety features
- Traffic lights - embedded systems can control timing sequences and respond to sensors

Non-Embedded Systems (General-purpose Computers):

- Smartphones
- Laptops
- Desktop computers